"Express Mail" mailing label number:

EV 335896396 US

MULTI-USER DATABASE SYSTEM AND METHOD FOR RESOURCE USAGE TRACKING

Mary B. Clark
Phil Lancour
Satwant Bal
Tom Stanek

FIELD OF THE DISCLOSURE

[1001] This disclosure is, in general, related to multi-user database systems and methods for resource usage tracking.

BACKGROUND

[1002] Large data warehouse systems are often used in decision and operational support systems. These large systems are typically accessible by multiple users through network interfaces or mainframe communications channels. Many such systems support parallel processing, and have multiple processors. Such processing capabilities permit many users to initiate sessions and request transactions with the data warehouse.

[1003] A problem arises with typical enterprise data warehouse systems when inefficient or erroneous requests are submitted. Inefficient requests may utilize excessive amounts of CPU time or input/output bandwidth. This excessive resource utilization may slow response times of the system or tax the resources available to other users. Erroneous queries compound the problem by utilizing CPU time while producing limited benefit to users. Typical enterprise data warehouses lack the ability to track and identify inefficient and erroneous transaction requests.

[1004] These typical enterprise data warehouse systems also lack the ability to track resource utilization based on particular user sessions and transaction requests. To forecast expected resource usage and to plan for resource allocation, it may be desirable

to have an historical record of resource utilization associated with session and request usage. However, typical enterprise data warehouses lack the ability to track resource usage such as CPU time and input/output traffic for unique sessions or transaction requests. Therefore an improved enterprise data warehouse system would be desirable.

SUMMARY

[1005] The disclosure is generally directed to a multi-user database system. The multi-user database system includes at least one processor, at least one network interface coupled to the processor, an event table, an accounting table, and a session table. The at least one network interface is coupled to the at least one processor. The at least one network interface is configured to receive transactions from a plurality of users. The transactions include session maintenance transactions and data requests. The event table stores a data log of the session maintenance transactions. The accounting table stores data associated with the data requests. The session table is derived from the event table and the accounting table. The session table stores resource usage data associated with at least one user session.

[1006] The disclosure is also directed to a multi-user database system. The multi-user database system includes a processor, a network interface, an event table, an accounting table, and a request table. The network interface is coupled to the processor. The network interface is configured to receive transactions from a plurality of users. The transactions include session maintenance transactions and data requests. The event table stores an event log of the session maintenance transactions. The accounting table stores data associated with the data requests. The request table is derived from the event table and the accounting table. The request table stores resource usage data associated with the data requests and associated with at least one user sessions.

[1007] The disclosure is further directed to a method of tracking database system usage. The method includes determining a set of new sessions from an event log data table to form a temporary sessions data table; matching entries in the temporary sessions data table with a set of request transactions to form a matched data table; preparing a sessions-level summary from the matched data table; updating a session table; and querying the

session table to track database systems usage. The session table stores resource usage data associated with the set of new sessions.

BRIEF DESCRIPTION OF THE DRAWINGS

[1008] **FIG. 1** depicts an exemplary embodiment of an enterprise data warehouse.

[1009] **FIG.s 2-A, 2-B and 2-C** depict an exemplary data flow.

[1010] **FIG. 3** illustrates an exemplary method for determining resource usage.

[1011] **FIG.s 4 and 5** depict exemplary data tables.

[1012] The use of the same reference symbols in different drawings indicates similar or identical items.

DESCRIPTION OF THE DRAWING(S)

[1013] FIG. 1 depicts an exemplary embodiment of an enterprise data warehouse 100. The enterprise data warehouse 100 may have one or more processors 102, network interfaces 104, programs 106, an accounting database 108, an event log database 110, a session information database 112, a sessions database 114, a history database 116, a request database 118, and various temporary databases 120. A processor or multiple processors 102 may interpret and perform transactions with the various databases and tables. Multiple processors may allow parallel processing and faster performance.

[1014] Various network interfaces 104 may provide access from remote locations to the enterprise data warehouse. From remote locations users may log on to the enterprise data warehouse 100 and perform transactions. These transactions may include session maintenance transactions such as opening a session or closing a session and database/table queries and maintenance transactions. Enterprise data warehouse 100 may also include various programs such as database engines, scripts, utilities, operating system code, and executables.

[1015] The enterprise data warehouse may have various databases or data tables such as an accounting table 108, an event log 110, a session info table 112, a sessions table 114, a history table 116, a request table 118 and various temporary tables 120. A database may include one or more tables with various logical links. Databases or tables may be alternately organized as more than one table. The accounting table 108 may track central processing unit (CPU) usage and input/output (IO) usage and associate these with an account name, a user name, an accounting string, or a request string. The CPU usage may, for example be expressed in processing time or clock units, among others. The IO usage may, for example, be expressed as bit or byte counts, packet counts, or usage time, among others. The event log 110 may, for example, track session maintenance transactions such as log-on and log-off transactions. The event log 110 may associate log-on dates and times with user names and account names and assign session identification numbers. The session information table 112 may track running sessions and associate these with a user name, an account name, or a session identification number. In some cases, third part software may be used to capture query strings. Data may be extracted from the accounting table 108, event log table 110, and session info table 112 to provide a session table 114, a history table 116, and a request table 118.

[1016] The session table 114 may provide a cumulative accounting of resource usage attributed to various sessions. The session table 114 may include entries for storing statistical characteristics and cumulative usage of CPU and IO usage, among others, associated with a given session. The request table 118 may hold entries that associate transaction requests with CPU and I/O usage statistics. The entries in the request table 118 may also be associated with user sessions.

[1017] The history table 116 may be used to indicate which transactions have been incorporated into the sessions table 114 and the request table 118. Periodically, the accounting table 108 may be purged. Between purges, the history table 116 may track transaction entries that have been processed into the session table 114 or the request table 118. In creating the sessions table 114 and the request table 118, various temporary databases 120 may be used.

[1018] In addition to the databases listed above, the enterprise data warehouse 100 may be used to store or access many other databases and tables. Furthermore, the enterprise data warehouse 100 may include various connections to other computational systems such as mainframes or other database systems. The enterprise data warehouse may also include software to capture query strings. One exemplary embodiment of the enterprise data warehouse 100 is a data warehouse by Teradata®.

[1019] FIG.s 2-A, 2-B, and 2-C depict a data flow useful in accumulating information from an accounting table 204, an event log table 206, and a sessions information table 254 to create a session table 240 and a request table 246. As shown in FIG. 2-A, a history table 202 may be queried with the accounting table 204 to determine which entries in the accounting table 204 have not been previously processed. The history table 202 may include a listing of those entries within the accounting table 204 that have been previously processed. The accounting table 204 may be periodically purged. If the periodic purging occurs less frequently than the processing of the transaction data, the history table 202 provides a listing of those transactions in the accounting table 204 that had been previously processed. The history table 202 may be purged at the same time as the accounting table 204. The query is performed as shown in transaction 212 to select the current transactions and store them in the current transactions accounting table 214. Once the current accounting transactions table 214 has been used in the process below, the history table 202 may be updated with an update history transaction 210.

[1020] The sessions table 208 may be used in conjunction with the event log 206 to determine which sessions are new or open. As shown in transaction 216, the event log 206 is compared to the session table 208 to determine a set of session entries that occurred after the last processing or maintenance. These may be stored in a "sessions since last maintenance" table 218.

[1021] As shown in FIG. 2-B, the current accounting table 214 may be compared with the "sessions since last maintenance" table 218. The current accounting table 214 stores information associated with CPU and I/O usage. The "sessions since last maintenance" table 218 stores data associated with new and open sessions. As shown with transaction

220, the tables may be matched based on user name and session identification. This comparison may yield a partially matched data table 222. Some entries within the current accounting table may not include a session identification field. In some records within the current accounting table 214, the accounting string will include session identification. These records may be parsed to determine the session identification. A second request 224 may be performed using the user name and account string on the partially matched data to provide a matched data set 226 having more matched data. The matched data set 226 may store transaction resource usage entries associated with session or request data. The request level statistics and session level statistics may then be summarized with a request level summary transaction 228 and a sessions-level summary transaction 230.

[1022] As shown in FIG. 2-C, the sessions-level summary transaction 230 results in a session update data table 232, and the request-level summary transaction 228 results in a request updates table 234. The session update table 232 may include sessions that were open as of the last processing and have new data or new sessions that were started after the previous processing. An update sessions transaction 236 may update the data associated with open sessions, and a insert sessions transaction 238 may insert new entries associated with new sessions occurring since the previous processing. These transactions 236 and 238 may be performed on a sessions table 240. Similarly, the request updates data table 234 may include open requests having new data and new requests. The update request transaction 242 may update request or transaction data in the request database 246 and the insert request transaction 244 may update the request table 246 with new entries associated with new requests or transactions.

[1023] In one exemplary embodiment, the event log 252 may be queried through transaction 248 to determine which sessions have been closed with log-off events. The session info database 254 may also be queried with transaction 250 to determine if any sessions are not running and close those sessions in table 240.

[1024] The sessions table 240 and the sessions table 208 may take the same format and represent the same table at different points in the data flow. In one exemplary embodiment, the sessions table 240 and the sessions table 208 are snap shots of the same

table at different points in time. Similarly, the event log 252 and the event log 206 may represent the same table at the same or different points in the data flow. In one exemplary embodiment, the event log 252 and the event log 206 are snap shots of the same table at different points in time.

[1025] FIG. 3 depicts an exemplary method for creating the sessions and requests table data. Further, this method may be used to determine historical trends in CPU and I/O usage and may also be useful in determining which transactions cause problems or result in excessive CPU and I/O usage, thus, enabling improved data warehouse management. As shown in step 302, a current set of accounting transactions may be determined. The current set of accounting transactions may include transactions not previously processed. In one exemplary embodiment, the current set of transactions may be determined by comparing an accounting database with a history table. A set of new and open sessions may also be determined as shown in step 304. The set of new and open sessions may, for example, be determined by comparing a session table with an event log. These two steps, 302 and 304 may occur simultaneously, or in various orders.

[1026] With the set of current accounting transactions and a set of new and open sessions, the data from these two sets may be matched on the user name and session identification as seen in step 306. In systems where the session identification may not be recorded with all entries, a further match may be performed using the user name and account string, as shown in step 308. In alternate embodiments, various matches may be performed in various orders. With these matches, a data table may be prepared from which a request summary may be drawn as seen in step 310 and a sessions summary may be drawn as seen in step 312.

[1027] With the summaries, session tables and request tables may be updated or new entries may be added. As shown in step 314, the session table may be updated with sessions that were previously opened and have new data. As shown in step 316, new sessions that were opened since the previous processing may be inserted into the sessions table. Similarly, the request table may updated as shown in step 318 or new requests may be inserted in the request table as shown in step 320. These steps, 314, 316, 318 and 320

may be performed in various orders or simultaneously, depending upon the capabilities of the enterprise data warehouse system.

[1028] The session table may be further updated by closing sessions that exhibit log-off events as shown in step 322. Further, the session table may be updated by closing sessions that are not running, as indicated by a session information table, as shown in step 324. If a history table is being used to monitor those transactions within the accounting table that have been previously processed, the history table may be updated as shown in step 326.

[1029] Once the sessions table and request table have been created, they may be queried to determine CPU and I/O usage, as shown in step 328. For example, the sessions table may be queried to determine the historical trend of CPU and I/O usage among various users. Such a trend may be useful in forecasting or allocating data warehouse usage and resources. In another example the request table may be queried to determine which transactions represent large uses of data warehouse resources. For example, transactions that utilize more than a select level of CPU or I/O bandwidth may be identified. In another example, transactions with relatively high resource utilization may be identified. Such queries may be useful in determining which transactions may be improved or may be monitored or modified to further improve data warehouse performance or reduce CPU and I/O usage. A query may also be useful in identifying problem transactions that may be corrected to enhance data warehouse performance. The MVS data fields and the MVS system or the source identification fields and the query capturing software may be used to identify inefficient transactions. As shown in step 330, code associated with inefficient transaction requests may be manipulated to improve database performance. For example, the code may be manipulated to request a smaller set of data or, the transaction may be deleted.

[1030] FIG. 4 depicts an exemplary session table. The table may include, for example, such fields as user ID, host ID, session ID, log-on date, log-on time, log-off date, log-off time, log-on performance group, log-off event, session duration, session CPU, session I/O, MVS data, IP address, process ID, account name, load data, load time, among others.

The user ID may identify a user. The host ID may identify a host. The session ID may provide a unique identification number to a session. The log-on date, log-on time, log-off date, and log-off time fields may provide temporal data associated with log-on and log-off events. However, other methods may be used to represent such data, such as using computer clock times that incorporate date and time. The log-on performance group field may be useful in ascertaining priority. The log-off event may include indications of whether the log-off was intentional or a timed-out event. The session duration field may present a total time or duration of session. The session CPU and session I/O fields may indicate CPU usage and input/output bandwidth usage. In addition to these fields, other fields may be included that incorporate statistics on the CPU and I/O or resource usage such as averages per transaction, maximum transaction usage, and minimum transaction usage.

[1031] Sessions may be manipulated or initiated from various sources. In some cases these sources may be a mainframe. As such, data associated with the mainframe transactions may be stored in the Multiple Virtual Systems (MVS) data fields. In one exemplary embodiment, the MVS data fields contain data that permits a user to look up a job on the MVS system to find more information about a transaction such as a query string. Other transactions may be manipulated through various network connections. As such, fields may be provided to record the source of these transactions such as where the source is identified by an IP address. In addition, software may be used to capture query strings from these other sources. The strings may then be identified using the source data. Further, the session table may associate other information with the session such as the process identification, the account names, and load dates and times. The load dates and times may indicate the date and time the entry was created or last updated.

[1032] FIG. 5 is an exemplary embodiment of a request table. A request table may include a host ID, a session ID, a user ID, log-on dates and times, request start date and request start times, request performance groups, access module processor (AMP) count, CPU and I/O data, parallelism data, ratios, and load dates and times. The host ID may identify the host. The session ID may provide a unique identification number for identifying the session. The user ID may identify the user. The log-on date and log-on

time may be associated with the session and may be stored in various formats, such as computer clock times. Similarly the request start date and request start time identify the start date and time of the request and may be stored in other formats. The request performance group identifies the priority provided to the request. In multiprocessor and parallel systems, a request may be performed by more than one processor or more than one process. The request table may also include data associated with how many processes are assigned to the request. This data may be stored in the AMP count. In addition, statistical data associated with CPU usage and I/O usage may be stored in various fields including the CPU and I/O fields. For example, average, minimum, and maximum resource usage per AMP may be stored in a related field. Other statistics such as parallelism and CPU-I/O ratio may be used in characterizing the performance of a transaction. The load date and time may be useful in determining when an entry was inserted into the table or updated.

[1033] The above disclosed subject matter is to be considered illustrative, and not restrictive, and the appended claims are intended to cover all such modifications, enhancements, and other embodiments which fall within the true spirit and scope of the present invention. Thus, to the maximum extent allowed by law, the scope of the present invention is to be determined by the broadest permissible interpretation of the following claims and their equivalents, and shall not be restricted or limited by the foregoing detailed description.